

DAFFER MCDANIEL

A LIMITED LIABILITY PARTNERSHIP

700 Lavaca, Suite 720
Austin, Texas 78701-3119

Intellectual Property Law
including Patents, Trademarks,
Copyrights and Unfair Competition

Phone (512) 476-1400
Facsimile (512) 703-1250

fax

Date: November 13, 2009

To: Examiner Isaac Tuku Tecklu Fax Number: 571-~~273~~-7957

From: Charles D. Huston

Comments: Following is the proposed agenda for the telephone interview scheduled on Monday, November 16, 2009 to discuss the differences between Tabloski and the amended claims.

DRAFT**PATENT**
5854-00100**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**Serial No. 10/700,254
Confirmation No. 3682

I hereby certify that this correspondence is being transmitted to the United States Patent & Trademark Office via electronic submission or facsimile on the date indicated below:

11/13/2009 /Pamela Gerik/
Date Pamela Gerik

AMENDMENT; RESPONSE TO OFFICE ACTION MAILED AUGUST 18, 2009

Dear Sir/Madam:

Responsive to the Office Action of August 18, 2009, please amend the case as follows:

Amendments to the Claims are reflected in the listing of claims which begins on page 2 of this paper.

Remarks begin on page 6 of this paper.

DRAFT

Amendments to Claims

This listing of claims will replace all prior versions and listings of claims in the application:

Listing of Claims

1. (currently amended) A system of managing data utilizing one or more processors and a single operating system in a multi-thread environment, comprising:

a plurality of map components, each map component having one or more ports for accepting data and for producing data and each map component encapsulating a particular dataflow pattern;

compiler tools for organizing and linking said map components using said ports into an executable dataflow application, said compiler tools, including a map synthesizer which validates links between map components and prepares an

DRAFT

3. (original) The system of claim 1, at least one map component having properties determining map component design behavior.
4. (original) The system of claim 1, at least one map component having properties that affect map component execution behavior.
5. (original) The system of claim 1, at least one of the map components comprising a composite component encapsulating a particular dataflow pattern using other map components as subcomponents.
6. (original) The system of claim 1, at least one of the map components comprising a scalar map component to process a specific data transformation.
7. (original) The system of claim 1, at least one of said ports linked to transfer specific types of data.
8. (original) The system of claim 1, at least one of said ports initially defined as a generic port for processing generic types of data, said generic port being later synthesized to transfer a specific sub-type of data.
9. (original) The system of claim 1, at least one of said ports being composite, comprising a plurality of hierarchical ports.
10. (original) The system of claim 1, at least one of said ports supporting multi-valued null data tokens.
11. (previously presented) The system of claim 1, at least one of said map components being encoded as an encrypted extensible markup language (XML) document.

DRAFT

12. (original) The system of claim 1, at least one of said map components being composite comprising a number of hierarchical dataflow graphs.

13. (currently amended) The system of claim 1, the compiler tools operating to remove design time links between map components to produce a flat dataflow graph containing a plurality of map processes for execution.

14. (canceled)

15. (currently amended) The system of claim 1, the ~~compiler tools synthesizer~~ operating to perform syntactic and semantic analysis, type inference and validation.

16. (currently amended) A method of transforming data in a multi-thread parallel processing environment comprising a single operating system and one or more processors wherein:

map components are assembled visually into an integrated dataflow application by

linking map processes within map components, wherein the map components are linked using ports;

the integrated dataflow application is executed in parallel by recognizing the linked map processes within the map components and allocating a separate thread to each of a plurality of linked map-process processes; and

each of said plurality of linked map process is executed on its allocated thread substantially in parallel as part of a single process, and said data resides in memory accessible to each of said plurality of linked map process.

17. (original) The method of claim 16, wherein said a-plurality of linked map processes read data tokens from input ports and write data tokens to output ports.

DRAFT

18. (currently amended) A method of managing data in a multi-thread environment, comprising:

accessing a library of map components at least some of said map components constituting a specific data transformation and having input and output ports;

assembling a dataflow application using map components from said library, wherein said map components each comprise one or more processes linked together using said ports; and

executing the assembled dataflow application with source data by assigning a separate thread to each a separate map component process where said threads execute as part of a single process in parallel on said source data without data partitioning.

19. (original) The method of claim 18, including imposing properties on the map components during assembly constraining the assemblage of the dataflow application.

20. (original) The method of claim 18, the map components including polymorphic ports which declare status as input and output ports during assemblage.

21. (previously presented) The system of claim 14, the executor operating on a single CPU in a hyperthread architecture.

22. (previously presented) The system of claim 14, the executor operating on a multiple processor core with at least some threads assigned to different processors.

23. (previously presented) The system of claim 14, the executor operating on multiple processors in a distributed network configuration.

24. (previously presented) The method of claim 16, communication between said processes executing in parallel being managed by an executor separate from the operating system.

DRAFT

25. (previously presented) The method of claim 18, including:

determining if a port will block execution of a thread; and

avoiding a deadlock by allowing the data queue to grow at said determined port.

26. (new) The method of claim 16, at least one map component being composite, thereby encapsulating a particular dataflow pattern.

DRAFT**REMARKS**

Claims 1, 2, 13, 15, 16, and 18 are amended, claim 14 is canceled, and claim 26 is added herein. Claims 1-13 and 15-26 are pending in the captioned case. Further examination and reconsideration of the presently claimed application are respectfully requested.

Section 102 Rejection

Claims 1-10, 12-14, and 16-24 were rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,999,729 to Tabloski, Jr. et al. (hereinafter "Tabloski"). The standard for "anticipation" is one of fairly strict identity. A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art of reference. *Verdegaal Bros. v. Union Oil Co. of California*, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987); MPEP 2131. Furthermore, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, as arranged in the claim. *W.L. Gore & Assocs. V. Garlock*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983). Using these standards, Applicant submits Tabloski fails to disclose each and every element of the currently pending claims, as set forth in more detail below.

Tabloski uses dataflow graphs (Tabloski -- Figs. 6, 7) to describe a parallel application, however Tabloski is a different type of system targeting a different problem than contemplated by the present invention. Tabloski targets a conventional parallel application where data is allocated or distributed to all the processors. There is no shared memory, the data is partitioned. Because Tabloski treats each "object" separately, there is minimal interprocess communication (Tabloski -- col. 20, line 30 – col. 24, line 36; Fig. 7). Tabloski uses a dataflow graph to generate code in a high level language (C++) (Tabloski -- col. 20, lines 15-29). This C++ code is then compiled at runtime into an executable program.

While a preferred form of the present invention includes a "compiler" it operates in a different manner and attains a different result than Tabloski. As shown in present Fig. 2A, this

DRAFT

“compiler tool” includes many design components, such as an assembler, loader, and synthesizer. In a preferred embodiment of the present invention, the compiler 12 (Tabloski -- Fig. 1) constructs a representation of the data flow graph where the composite operators are in-lined and the ports configured to validate and optimize the graph prior to execution. At execution, this graph representation is used to load and configure the map components, link the ports and to start each map component execution on a separate thread as part of a single process. The claims as amended reflect these different constructs for “compiler tools.” Such “compiler tools” have no counterpart in Tabloski.

Tabloski does not execute map processes on a separate thread. Instead, Tabloski executes objects concurrently as separate processes -- not in a shared memory architecture. There are no composite components or composite maps or composite or hierarchical ports. Therefore, it is not surprising that in Tabloski, the execution objects communicate blockage to an execution control object (a processor).

In a preferred form of the present invention, applications are targeted at shared memory, multiprocessing (SMP) systems and executed on threads in a single process image. Data is transferred between operators using queues via the ports. Tabloski achieves parallelism by executing objects concurrently as separate processes (Tabloski -- col. 20, line 30 -- col. 24, line 36; Fig. 7). Tabloski uses an execution control object on each node of the networked processing environment to communicate with the execution objects.

Turning to the claims, because Tabloski does not use threads to execute objects in the dataflow, it is not surprising that Tabloski fails to teach or disclose “assigning a separate thread to each map process” as described in claim 1. Indeed, Tabloski does not mention the notion of “threads” or a multiple thread environment. See, e.g., present claim 16 (“allocating a separate thread to each of a plurality of map processes”) and present claim 18 (“assigning a separate thread to a separate map process . . .”). Support for the notion of assigning a separate thread to separate map processes can be found through out the present application, e.g., pg. 4, lines 1-19; pg. 8, lines 10-15, pg. 29, lines 16-22; pg. 30, lines 1-5.

DRAFT

Tabloski fails to teach or disclose the use of composite operators or composite ports. See, e.g., present claims 5, 7, 12, 26. As outlined above, Tabloski fails to disclose the “compiler tools” as described in present claim 1. These and other differences are apparent in the claims currently presented and Tabloski.

For at least the reasons stated above, Applicants assert independent claims 1, 16, and 18, as well as claims dependent therefrom, are not anticipated by Tabloski. Accordingly, removal of this rejection is respectfully requested.

Section 103 Rejection

Claim 11 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Tabloski in view of U.S. Patent No. 7,095,852 to Wack et al. (hereinafter “Wack”). Claim 15 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Tabloski in view of U.S. Patent No. 6,993,753 to Yamanaka (hereinafter “Yamanaka”). Claim 25 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Tabloski in view of U.S. Patent No. 7,124,405 to Kakivaya et al. (hereinafter “Kakivaya”).

For at least the same reasons discussed above regarding the patentability of independent claims 1 and 18, Applicants believe claims 11, 15, and 25 dependent therefrom are also patentable over Tabloski or the combinations of Tabloski and Wack, Yamanaka, or Kakivaya. Furthermore, because Tabloski presents a different solution and architecture in a distributed data parallel application, it is not seen how Tabloski can reasonably be applied. Tabloski would teach away and a predictable result would not be taught. *DePuy Spine v. Medtronic*, 567 F.3d 1314 (Fed. Cir. 2000). Accordingly, Tabloski does not appear to be applicable under § 103. Accordingly, removal of this rejection is respectfully requested.

DRAFT**CONCLUSION**

The present amendment and response is believed to be a complete response to the issues raised in the Office Action mailed August 18, 2009. In view of the amendments and remarks herein, Applicants assert that pending claims 1-13 and 15-26 are in condition for allowance. If the Examiner has any questions, comments, or suggestions, the undersigned attorney earnestly requests a telephone conference.

No fees are required for filing this amendment; however, the Commissioner is authorized to charge any additional fees which may be required, or credit any overpayment, to Daffer McDaniel, LLP Deposit Account No. 50-3268.

Respectfully submitted,

/Charles D. Huston/
Charles D. Huston
Reg. No. 31,027
Attorney for Applicant(s)

Customer No. 35617
Date: November 13, 2009